# Development of an Improved ResNet-18 for CIFAR-100 Image Recognition

Vajra Keller, MASc

**Queen's University**
Department of Mechanical and Materials Engineering

## Abstract

*Residual connections are beneficial for neural networks because of shortcut projections that can skip over one or more convolutional layers. These connections allow alternative avenues for learning to occur while limiting performance degradation associated with deeper networks. In this study, ResNet-18 was explored as a benchmark for image recognition on a CIFAR-100 dataset. The ResNet-18 architecture was systematically modified to improve the model's performance, including some changes to the model's hyperparameters. Pre-training, 3x3 residual convolutions, and using an Adam optimizer with gradient clipping resulted in substantial improvements to the base model. Adding dropouts resulted in worse accuracy compared to the base model and pre-activation (BN-ReLU-Conv) did not greatly affect performance.*

## 1. Introduction

Image recognition refers to the semantic separation of images into different groups or classes. Since LeNet's [1] inception in 1998, large scale image recognition tasks have remained a relevant challenge for convolutional neural networks. Humans can naturally perform this task very well, but recent developments in computer vision and deep learning techniques have shown that computers can achieve similar levels of performance to humans [2].

In supervised machine learning models of this task, the input image label is iteratively compared against the expected output label. Traditional convolutional neural networks are typically comprised of convolutional layers that perform feature extraction and a fully-connected layer at the end for image classification. During training, the gradients or "weights" at each layer are learned using an optimization algorithm with backpropagation. The performance of the model depends largely on these weights and how well the weights can generalize to unseen datasets.

Between 2010 and 2017, convolutional neural networks developed for the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) unveiled some elegant and novel approaches for improving image recognition tasks on complex, multi-class datasets. Some notable contributions include networks such as AlexNet [3], which was the first to implement data augmentation, ReLU [4], and dropouts [5], and ResNet [6], which builds upon previous networks such as VGG [7] and GoogLeNet [8].

The purpose of this study was to carry out a systematic exploration on residual networks. To achieve this, ResNet-18 was implemented from scratch and the base model subjected to 12 different architectural and hyperparameter modifications to produce a final, best performing convolutional neural network. ResNet-18 was chosen as it offers a good balance between training time, performance, and network depth. The training loss, validation loss, and top-1 accuracy performance for each modification are presented here after training and testing on a CIFAR-100 dataset.

## 2. Background

### 2.1 Deeper Networks

A common and intuitive approach for improving model performance is to increase the depth of the network [9]. Deeper networks with more hidden layers have increased complexity due to the larger number of trainable parameters within the network. [7] showed that small convolutional layers can be stacked together to obtain the same size receptive field as a single larger convolutional layer, while boosting performance due to the increase in number of non-linear activations between layers.

However, deeper networks do not necessarily lead to better performance as they can run into network degradation issues related to low convergence and overfitting, resulting in poor model performance during testing [6][10][11]. Moreover, increasing the number of

layers adds to the computational costs of the model, which means even longer training times.

Overcoming these issues involves careful consideration of the network architecture and adjustments to the model's hyperparameters.

### 2.2 Residual connections

[6] presented a novel approach to combat network degradation in very deep networks, showing remarkable performance in models consisting of up to 1000 layers. Network degradation is limited by residual or "skip" connections from one layer to a few layers downstream of the network, as shown in Fig. 1 [6]**.** The outputs are then added together in element-wise fashion.

In extreme cases, if the learned feature map of intermediate convolutional layers has near zero gradients, residual connections can help boost network performance by preserving the learned gradients from previous layers.

Performance in a residual network is expected to be better (if not equal) to performance of a similarly sized plain network. The best performing network in [6] won first place in ILSVRC 2015, achieving a top-1 accuracy of 80.62% on ImageNet and 93.57% on CIFAR-10 datasets.

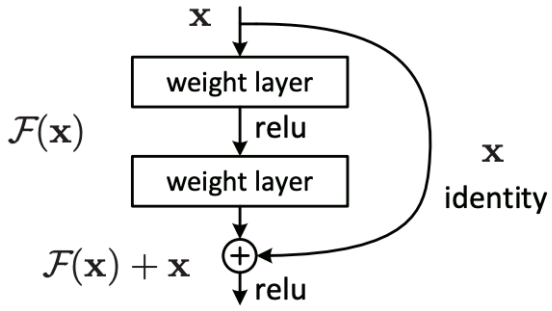


Figure 1. A residual learning building block featuring "skip" connections. The output of x is added in element-wise fashion to F(x), preserving previous layer features.

### 2.3 CIFAR-100

CIFAR-100 is a complex dataset with 20 superclasses subcategorized into 100 classes containing 600 images each [12]. Each class is made up of 500 training images and 100 testing images. These images have dimensions of 32x32 pixels and 3 color channels. The first 10 images from each superclass are shown in Fig. 2.

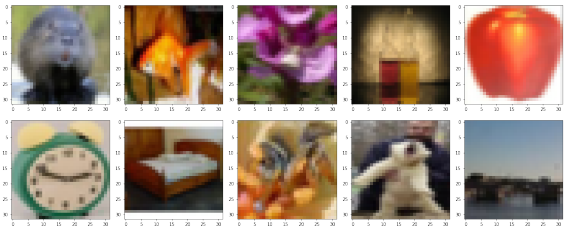

Figure 2. The first image from the first 10 superclasses in CIFAR-100. From left to right (classes): Beaver, aquarium fish, orchid, bottle, apple, clock, bed, bee, bear, and bridge.

Images in the dataset were digitally augmented to artificially increase variety in the training dataset following [3]. Specifically, the original image and its horizontal flip were randomly cropped and normalized according to its per pixel mean. Test images were normalized without performing any additional augmentations. This method of data augmentation was used in all of the network implementations explored in this study. An example of the input images in a training mini-batch with data augmentation is shown in Fig. 3.

## 3. Experiments

All of the experiments performed in this study were implemented using PyTorch, an open-source machine learning framework, and run on GPUs on Google Colaboratory Pro (Google LLC, Mountain View, CA).

### 3.1 ResNet-18

ResNet-18 was implemented from scratch to facilitate modifications to its network-level components. Where appropriate, the ResNet-18 architecture was verified against the original authors' implementation of ResNet-18 [6], ensuring that the hyperparameter configuration, connection between layers, and number of trainable parameters were correct. In particular, the original paper utilized a mini-batch size of 256, an initial learning rate of 0.1, a weight decay of 0.0001, and a momentum of 0.9. Fig. 4 shows a block diagram with output projections of the original ResNet-18 architecture alongside some of the modifications explored in this study. The original model was trained using stochastic gradient descent (SGD) with backpropagation.

In total, ResNet-18 is comprised of 17 convolutional layers and one fully-connected layer. There are an additional three convolutional operations in the residual

connections which performs resolution down-sampling of the input while up-sampling channel depth. All networks were trained for 25 epochs.

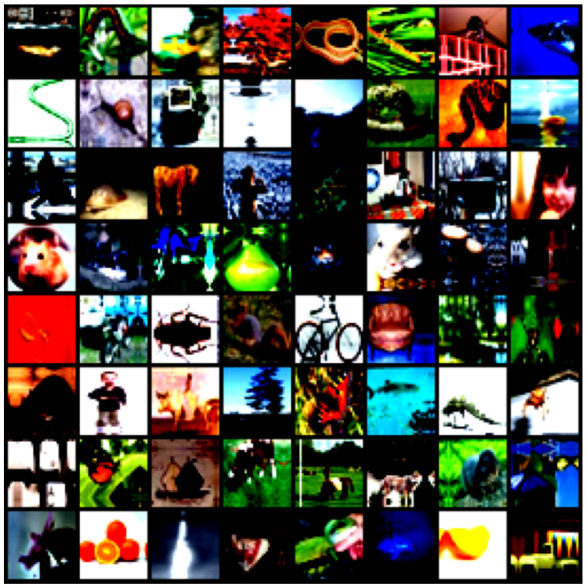

Figure 3. Randomly cropped and normalized images used for training all of the network iterations in this study.

### 3.2 Decreasing Initial LR (0.1, 0.01, 0.001)

The first experiment investigated the effect of initial learning rate (LR) on SGD with backpropagation. The LR is a hyperparameter which determines the step size at each iteration while moving towards minimizing the cross-entropy loss function. This is similar to tuning the speed at which the network learns, with larger LRs increasing the rate of convergence at the cost of potentially overshooting gradients around local minima.

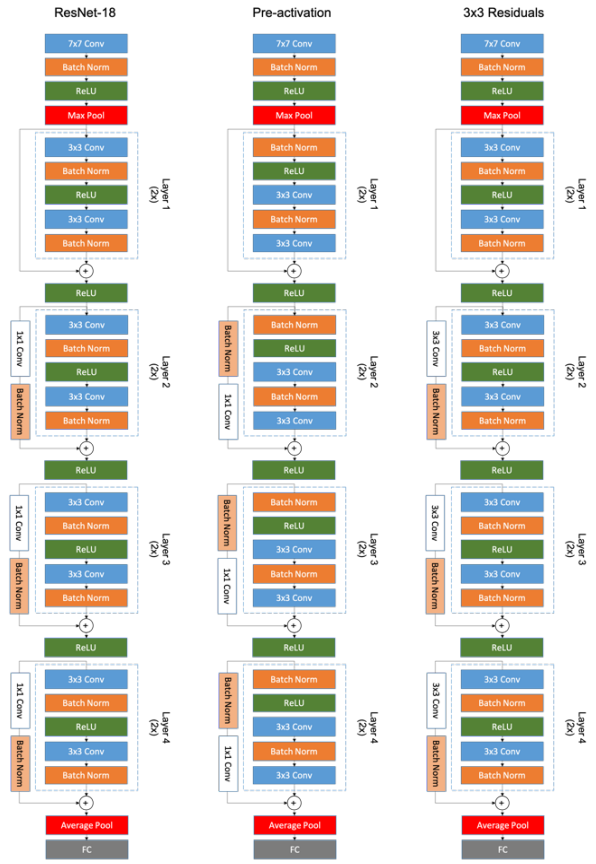


Figure 4. ResNet-18 architecture and some of the modifications explored in this study. **Left:** base ResNet-18. **Middle:** Pre-activation (batch normalization and ReLU preceding convolutional layer operations). **Right:** implementation of 3x3 convolutional layers in ResNet-18 residual connections.

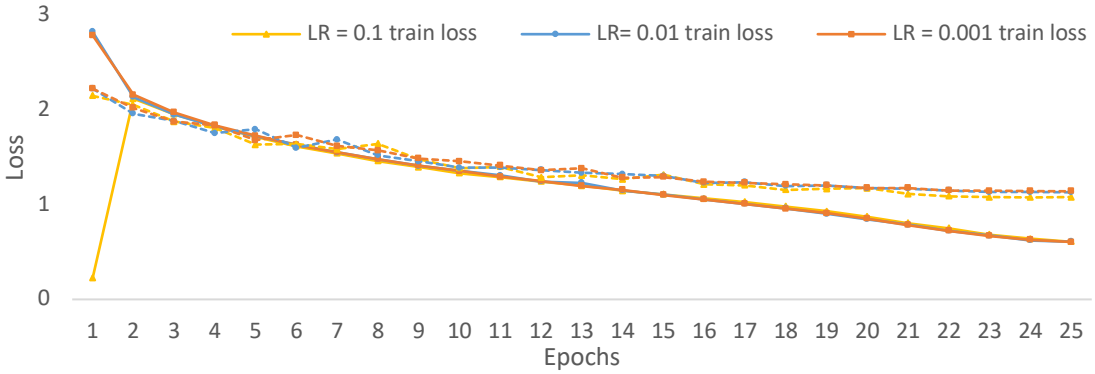


Figure 5. Impact of 0.1, 0.01, and 0.001initial learning rate (LR) on ResNet-18 training and validation losses. These modifications resulted in similar training and validation losses, but a learning rate of 0.1 achieved the best performance in terms of accuracy.

Three initial LRs were explored: 0.1 (like original), 0.01, and 0.001. In all of the experiments, a "one cycle policy" [13] was adopted to cyclically modulate the LR for each batch during training.

It was found that all three initial LRs exhibited similar training and validation losses (Fig. 5), but an initial LR of 0.1 performed the best in terms of accuracy (Table 1). Initial LR did not greatly impact computation time of the network.

### 3.3 Adam Optimizer and Gradient Clipping

This experiment looked at the impact of Adam optimizer and gradient clipping as hyperparameters for the network. Adam [14] is an adaptive moment estimation algorithm typically used in place of SGD for minimizing the error rate during learning. Adam is said to be fairly robust to hyperparameter choice and converges faster than SGD [14] but it may also lead to lower generalizability of the network [15].

Additionally, gradient clipping can be used for tackling exploding gradients by rescaling gradients that become too large into a smaller range [16]. Used in conjunction with Adam and a one cycle policy scheduler, gradient clipping may improve the generalizability of the network.

Results from this experiment show that using Adam alone yields higher training losses compared to SGD but it also increased accuracy from 66.35% to 68.25% (Fig. 6; Table 1). When gradient clipping is used in conjunction with Adam, training losses were lower than SGD and only a modest improvement to accuracy (66.47%) was seen. Computation time of using Adam added nearly 30 seconds compared to the SGD model.

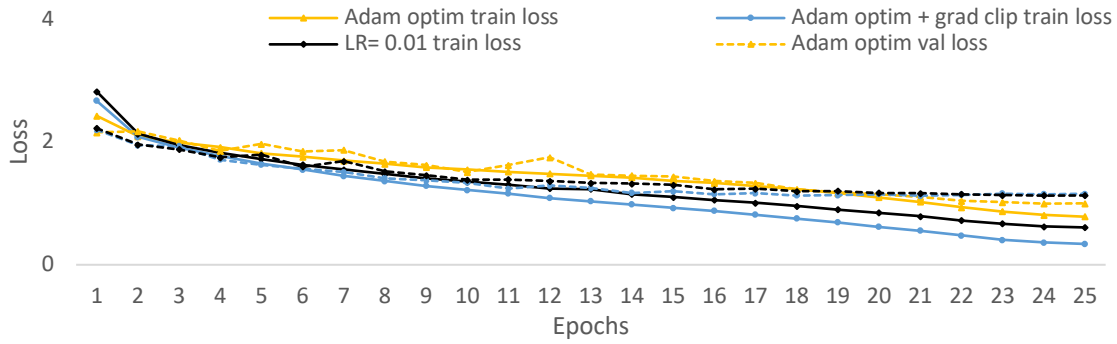


Figure 6. Impact of Adam optimizer and gradient clipping (grad clip) on training and validation losses. Using Adam optimizer results in more training loss at 25 epochs, but combined with gradient clipping, led to fewer losses compared to stochastic gradient descent (SGD). Accuracy of Adam optimizer with gradient clipping was higher than ResNet-18 on SGD, but lower compared to Adam optimizer alone (Table 1).

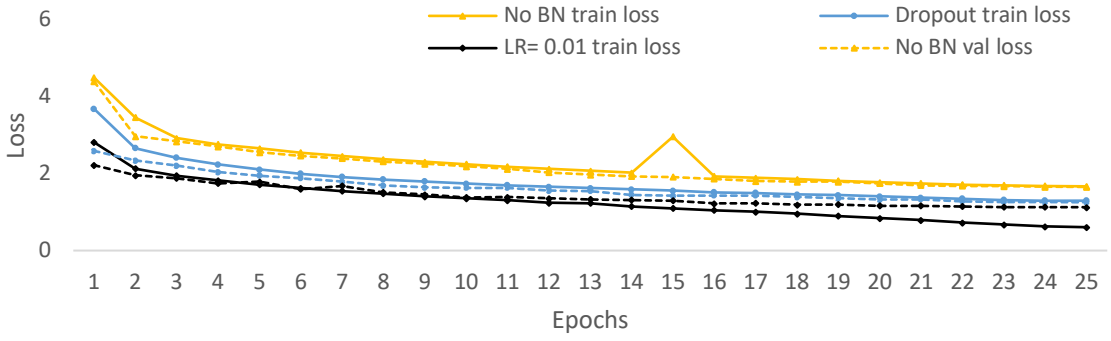


Figure 7. The impact of removing batch normalization or adding dropouts on training and validation losses. Removing batch normalization everywhere led to higher training and validation losses, but the network did not seem to overfit more compared to the standard ResNet model with an initial LR = 0.01. Removing batch normalization decreased accuracy to 48.5%. Adding dropouts decreased accuracy to 59.98%.

### 3.4 Removing Batch Normalization and Adding Dropouts

Regularization techniques such as normalized initialization and batch normalization are commonly employed in deep learning networks to reduce overfitting. These methods involve normalizing the pixel value of the input space to be centered around a mean of 0 and a standard deviation between 1 and -1 [17]. Normalization improves network performance because it prevents strong inputs from dominating the learned features of the network and increases the representation of otherwise weak features to be learned [17]. In [6], batch normalization was implemented after every single convolutional layer in the network.

In this experiment, the effects of completely removing all batch normalization processes in the network were observed. Removing batch normalization resulted in higher training and validation errors (Fig. 7), and greatly decreased model accuracy from 66.35% to 48.5% (Table 1). No significant overfitting was observed.

Adding a dropout with a probability of 0.2 in each residual block and a dropout probability of 0.5 before the final fully-connected layer caused the final epoch validation accuracy to decrease to 59.98% compared to the base model. This modification did not show significant divergence between training loss and validation losses after 25 epochs (Fig. 7).

### 3.5 Wider Residual Blocks, 3x3 Residual Convolutions, and Pre-Activation (BN-ReLU-Conv)

We then explore architectural changes to the base ResNet-18 model to improve its top-1 accuracy. The first modification increases the width of the residual blocks, implementing a k-value of 2 similar to [18]. The width of each residual block was increased from 64, 128, 256, and 512 to 128, 256, 512, and 1024 channels respectively.

Increasing the width of the residual blocks increased the total number of trainable parameters from 11,227,812 to 44,695,716. For an almost four-fold increase in number of parameters, training time increased by 81 seconds (Table 1). Accuracy of the model improved from 66.35% to 68.17%.

Another modification considered was replacing the 1x1 convolutions that perform identity mapping in the skip connections with 3x3 convolutions.

Replacing 1x1 convolutions with 3x3 convolutions increased the total number of trainable parameters from 11,227,812 to 12,604,068. This change resulted in lower training losses compared to the base model, but validation losses were similar (Fig. 8). The final epoch accuracy improved over the base model to 67.88%. Surprisingly, computation time was 168 seconds faster, but this may be due to Google Colaboratory GPU throttling at training time.

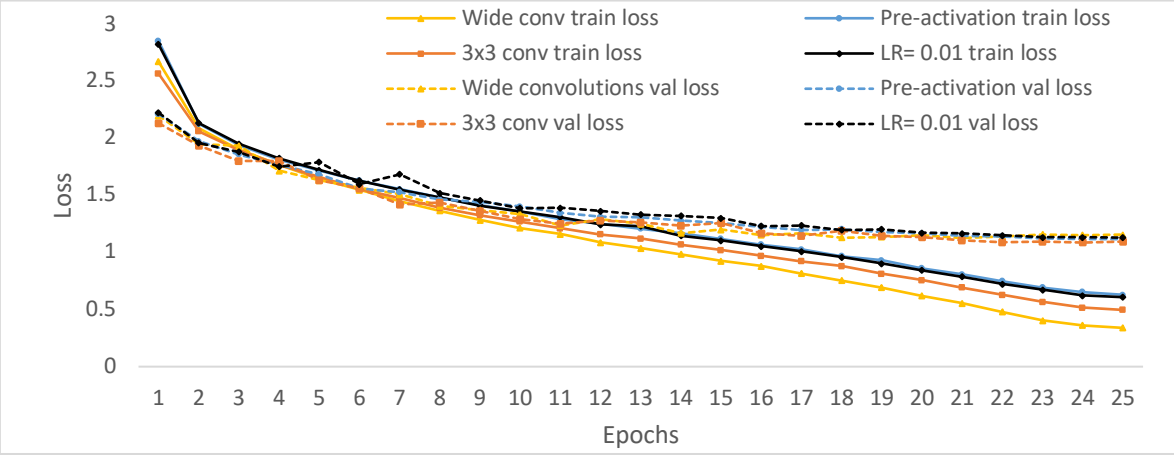


Figure 8. Effects of using wider residual blocks (wide conv), 3x3 residual convolutions (3x3 conv), and pre-activation on training and validation losses. Wider residual blocks and 3x3 residual convolutions improved accuracy to 68.17% and 67.88%, respectively. Pre-activation led to a marginal loss in accuracy (66.09%).

The last architectural modification investigated the impact of reordering the batch normalization, ReLU, and convolution components into a "pre-activation" topology introduced by [19] (refer to Fig. 4). This pre-activation scheme was implemented at each residual block. Batch normalization was also implemented before 1x1 convolutions in the residual connections.

Implementing the pre-activation topology resulted in no noticeable improvement in terms of losses compared to the base model. Validation accuracy at the final epoch was 66.09%, which is only marginally worse compared to the base model's accuracy of 66.35%.

### 3.6 New Wide ResNet-18

The results of the previous experiments were used to inform the design of an improved ResNet-18 architecture. This revised ResNet-18 network, dubbed "New Wide ResNet", combines the observed advantages of hyperparameter modifications such as using Adam optimizer with gradient clipping, and incorporates architectural changes with wider residual blocks and 3x3 convolutions in the residual connections. This implementation has 50,266,276 total trainable parameters. A mini-batch size of 256, an initial LR of 0.01, and a one cycle policy scheduler was incorporated into the model.

Combining these modifications resulted in a model bolstering significant improvements in final epoch validation accuracy over the base ResNet-18 model (72.51% vs 66.35%). While training losses were observed to be similar to the base model, validation losses of New Wide ResNet were lower in comparison (Fig. 9). New Wide ResNet also appears to generalize better compared to the base model. As a consequence of the larger number of total parameters, New Wide ResNet took 187 seconds longer to train.

### 3.7 ResNet-18 with Transfer Learning

As a final point of comparison, the New Wide ResNet was compared against a base ResNet-18 network pre-trained on ImageNet. This pre-trained network was imported from PyTorch's model library for ResNet-18. The hyperparameters for this model were kept similar to the original paper's implementation, namely an initial LR of 0.01, a mini-batch size of 256, and SGD with backpropagation. A one cycle policy scheduler and no gradient clipping was implemented. The pre-trained model was re-trained on the CIFAR-100 dataset without freezing any of the weights.

Interestingly, the pre-trained ResNet-18 model yielded better accuracy compared to the New Wide ResNet model (73.26% vs 72.51%). Because the pre-trained ResNet-18 model architecture is identical to the base ResNet-18 network, it also has much fewer parameters compared to the revised network. Training time of the pre-trained network was 350 seconds faster compared to our model.

The pre-trained model, however, shows signs of overfitting as the training losses and validation losses start to diverge after only 6 epochs. Meanwhile, our New Wide ResNet appears to be much more robust to overfitting.

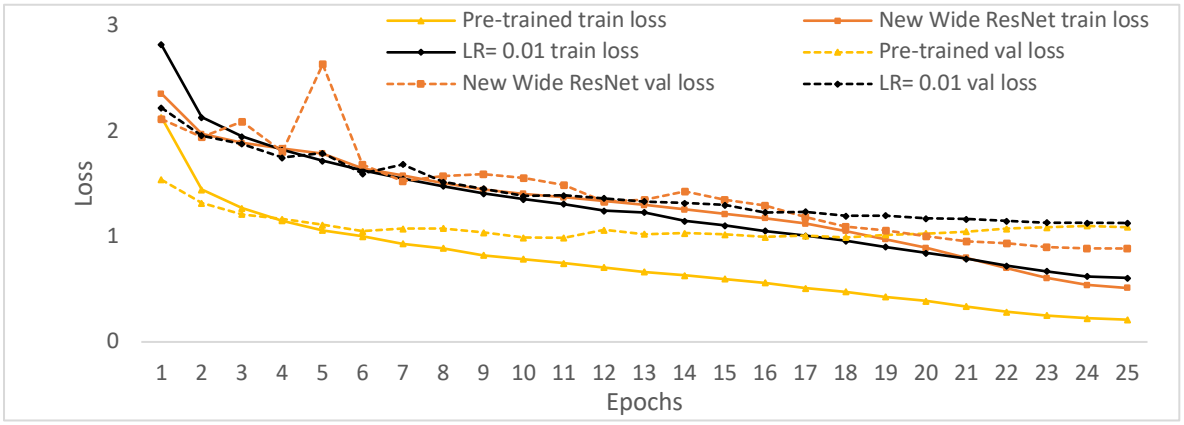


Figure 9. Training and validation losses of New Wide ResNet-18, pre-trained ResNet-18, and base ResNet-18. New Wide ResNet losses has better generalizability compared to the other models and performs slightly worse in accuracy compared to the pre-trained model.

| Method | Total parameters | Training time (25 epochs) | Top-1 Accuracy |
|---|---|---|---|
| ResNet-18 LR = 0.1 | 11,227,812 | 15min 8s | 0.6820 |
| ResNet-18 LR = 0.01 | 11,227,812 | 15min 14s | 0.6635 |
| ResNet-18 LR = 0.001 | 11,227,812 | 15min 7s | 0.6594 |
| ResNet-18 Wide residual blocks | 44,695,716 | 16min 35s | 0.6817 |
| ResNet-18 3x3 conv res | 12,604,068 | 12min 26s | 0.6788 |
| ResNet-18 Pre-activation | 11,226,020 | 15min 19s | 0.6609 |
| ResNet-18 w/o BN | 11,218,212 | 15min 7s | 0.4850 |
| ResNet-18 w/ Dropout | 11,227,812 | 14min 37s | 0.5998 |
| ResNet-18 Adam optimizer | 11,227,812 | 15min 46s | 0.6825 |
| ResNet-18 Adam + Grad clip | 11,227,812 | 15min 45s | 0.6647 |
| ResNet-18 Pre-trained | 11,227,812 | 12min 31s | **0.7326** |
| New Wide ResNet-18 | 50,266,276 | 18min 21s | 0.7251 |
| Wide Residual Network-16 (k = 8) [18] | | | 0.7957 |
| ResNet-110 [6] | | | 0.7458 |

Table 1. Summary of modifications, number of trainable parameters, training time (25 epochs), and top-1 accuracy of the experiments in this study. The top-1 accuracy of the best performing model explored is highlighted in bold, which was achieved by ResNet-18 model pre-trained on ImageNet. The second-best performance was obtained by New Wide ResNet-18.

## 4. Discussion

The objective of this study was to investigate the effects of different architectural and hyperparameter modifications to the ResNet-18 model implemented by [6] to inform the design of a final, improved deep learning model. To this extent, 12 modifications were explored in this study including changes to initial learning rate, optimizer function, topology, dropouts, and pre-training.

The first experiment looked at whether decreasing the initial learning rate from 0.1 to 0.01 and 0.001 had an impact on model performance. Except for these conditions, the network architecture and other hyperparameters such as mini-batch size, weight initialization, and SGD with backpropagation were identical. This experiment revealed that initial learning rate did not have a significant impact on performance. This may be likely due to the implementation of a "one cycle policy" scheduler in these models which modulates the learning rate for each batch. Therefore, regardless of the initial learning rate, the scheduler cyclically varies the learning rate at each iteration to facilitate the process of training and validating the network. Cyclical learning rates were introduced by [20] and discussed in [21] as an improvement over conventional methods of grid or random search, which are more computationally expensive. For simplicity, an initial learning rate of 0.01 with a one cycle policy scheduler was opted for testing the remainder of the experiments.

We also investigated the impact of using an Adam optimizer with gradient clipping over SGD on model performance. Replacing SGD with Adam alone increased validation accuracy by almost 2%, but when in conjunction with gradient clipping, only marginally improved accuracy.

The authors describe Adam as a combination of an adaptive gradient algorithm that improves performance on problems with sparse gradients and an algorithm that adjusts according to the average of recent magnitudes of the gradients [14]. Adam would be a suitable choice for training networks on sparse and noisy data like CIFAR-100, which is made up of low 32x32 resolution images to begin with. Gradient clipping may therefore hamper Adam's functionality because it relies on the momentum from previous gradients to find a suitable parameter learning rate, whereas gradient clipping limits the size of the gradients to be bound within a certain range.

In our experiments we found that removing batch normalization and adding dropouts negatively impacted model accuracy. The effect of removing batch normalization is not surprising, given that the function of batch normalization would ensure that the gradients in the feature maps are normalized according to the statistics of the batch. This improves the generalizability and efficiency of the network by increasing the strength of weak representations and reducing the impact of strong, dominating inputs [17].

Surprisingly, adding dropouts worsened our model's performance. In [5], dropouts are supposed to reduce overfitting because it reduces the reliance on any single

connection and has the effect of reinforcing neighboring neurons. Typically, dropouts are used for fully-connected layers. In our case, including dropouts at every convolutional layer may not have contributed to this desired effect.

Our next set of experiments attempt to modify the network architecture by introducing wider residual blocks, replacing 1x1 convolutional layers in the residual connections with 3x3 convolutions, and implementing a "pre-activation" topology in our network.

Increasing the channel depth at each residual block likely resulted in better accuracy because it increased the number of learnable features at each layer. Similarly, replacing 1x1 convolutions with 3x3 convolutions has the benefit of increasing the size of the receptive field at the residual connections. Instead of performing identity mappings (in the case of 1x1 convolutions in the base network), 3x3 convolutions can also play a role in feature extraction if learning in the residual blocks is suboptimal. However, this also introduces more complexity into the network, and this was observed at the expense of computational cost.

Modifying the network into a pre-activation topology has been explored in more detail by [19]. In a pre-activation topology, batch normalization and the nonlinear activation function precedes the convolutional layer. This is said to facilitate the training process by introducing regularization before the learning step and therefore helps reduce overfitting. However, our study found that this configuration did not greatly impact model performance.

With these results in mind, the New Wide ResNet network was implemented using an Adam optimizer with gradient clipping, wider residual blocks and 3x3 convolutions in the residual connections. Our model was able to perform well, achieving a top-1 accuracy of 72.51% on CIFAR-100, but it did not perform better than a ResNet-18 that has been pre-trained on ImageNet. The most reasonable explanation for this would be that a network trained on a larger, more varied dataset will tend to have better generalizability to new data. The quickest way to improve New Wide ResNet performance maybe to train it beyond only 25 epochs, as our model already shows less overfitting compared to the pre-trained network. If relying only on the CIFAR-100, data augmentation without replacement and testing on varying image scales like in [7] may improve performance even more.

## 5. Conclusion

We have explored 12 modifications to the base ResNet-18 network with the goal of implementing a new, improved network. One large limitation of this study is the lack of actual quantitative analyses into how each modification impacts the model. Future experiments can look at using deconvolutional layers to visualize the output of each feature map and statistical comparisons to observe the variance of the outputs.

## References


[1] Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86(11), 2278-2324.*

[2] Geirhos, R., Janssen, D. H., Schütt, H. H., Rauber, J., Bethge, M., & Wichmann, F. A. (2017). Comparing deep neural networks against humans: object recognition when the signal gets weaker. *arXiv preprint arXiv:1706.06969*.

[3] Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1* (pp. 1097–1105). Curran Associates Inc..

[4] Nair, V., & Hinton, G. E. (2010, January). Rectified linear units improve restricted boltzmann machines. In *Icml*.

[5] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929-1958.

[6] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[7] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[8] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

[9] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798-1828.

[10] Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Training very deep networks. *arXiv preprint arXiv:1507.06228*.

[11] Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

[12] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images (Technical Report). University of Toronto.

[13] Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1--learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*.

[14] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[15] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[16] Zhang, J., He, T., Sra, S., & Jadbabaie, A. (2019). Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881*.

[17] Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). PMLR.

[18] Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

[19] He, K., Zhang, X., Ren, S., & Sun, J. (2016, October). Identity mappings in deep residual networks. In *European conference on computer vision* (pp. 630-645). Springer, Cham.

[20] Smith, L. N. (2015). No more pesky learning rate guessing games. CoRR, abs/1506.01186, 5.

[21] Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1--learning rate, batch size, momentum, and weight decay. arXiv preprint arXiv:1803.09820